

1
2
3
4
5 Description

6
7
8
9
10 **GENERIC DETECTION AND ELIMINATION OF MACRO VIRUSES**

11 Technical Field

12 This invention pertains to the field of detecting and
13 eliminating computer viruses of a particular class known as macro
14 viruses.
15

16 Background Art

17 U.S. patent 5,398,196 discusses the detection of viruses
18 within a personal computer. However, unlike the present invention,
19 this reference does not treat the elimination of detected viruses,
20 nor does it discuss macro viruses.
21

22 A first existing technology used by anti-virus programs to
23 detect and repair macro viruses requires, for each unique new macro
24 virus, the development of a detection and repair definition (virus
25 signature). Thus, this first technology is capable of detecting
26 only publicly identified macro viruses (see below definition).
27 After the development of the detection and repair definition, the
28 user's anti-virus software must be augmented with the new
definition before it can detect the newly discovered macro virus.
This method has the advantage that a skilled anti-virus researcher
is able to study the virus and understand it enough so that a

1 proper detection and repair definition can be created for it. The
2 main disadvantage is that a relatively long turnaround time is
3 required before the general public is updated with each new
4 definition. The turnaround time includes the duration during which
5 the virus has a chance to spread and possibly wreak havoc, the time
6 to properly gather a sample and send it to an anti-virus research
7 center, the time required to develop the definition, and the time
8 to distribute the definition to the general public. This process
9 is similar to the process used for protecting against the once more
10 prevalent DOS-based viruses.

11
12
13
14 A second technology uses rudimentary heuristics that can scan
15 for newly developed macro viruses. These heuristics employ expert
16 knowledge of the types of viruses they seek. Often these
17 heuristics look for strings of bytes that are indicative of viral
18 behavior--for example, strings found in currently known viruses.
19 Current heuristics are very good at detecting, with high level of
20 confidence, new viruses that are variants of known viruses, but not
21 so good at detecting new viruses that are not variants of known
22 viruses. Another disadvantage of most current heuristics is that
23 they are good for viral detection only, and are not capable of
24 eliminating macro viruses once found. This is true of both macro
25 virus heuristics and DOS virus heuristics. The present invention
26 is an example of the second technology (heuristics). It is capable
27
28

1 of identifying both publicly identified macro viruses and publicly
2 unidentified macro viruses (see the below definitions).
3
4 Furthermore, it offers the significant advantage that it is capable
5 of eliminating macro viruses that are detected.

6 Disclosure of Invention

7
8 A computer implemented method, apparatus, and computer
9 readable medium for detecting macro viruses in code (15) adapted
10 for use on a digital computer (1). A detection module (17)
11 analyzes the code (15) to determine whether the code (15) contains
12 instructions causing a macro (8) to be moved to a global
13 environment (13), and whether the code (15) also contains
14 instructions causing the same macro (8) to be copied to a local
15 document (11). When these two conditions are satisfied, detection
16 module (17) declares that a macro virus is present within the code
17 (15).
18
19

20 Brief Description of the Drawings

21
22 These and other more detailed and specific objects and
23 features of the present invention are more fully disclosed in the
24 following specification, reference being had to the accompanying
25 drawings, in which:
26
27
28

1 Figure 1 is a block diagram showing the type of application
2 program 5 in the existing body of art that can be contaminated by
3 macro viruses detectable by the present invention.
4

5 Figure 2 is a block diagram showing global environment 13
6 associated with application program 5 of Figure 1.
7

8 Figure 3 is a block diagram showing how a macro virus can
9 contaminate the computing environment illustrated in Figures 1 and
10 2.
11

12 Figure 4 is a block diagram showing a preferred embodiment of
13 the present invention.
14

15 Figure 5 is a flow diagram showing a preferred embodiment of
16 the present invention.
17

18 Definitions

19 As used throughout the present specification and claims, the
20 following words and expressions have the indicated meanings:
21

22 "macro" is a computer program written using a structured
23 programming language and created from within an application program
24 that has a global environment and can create local documents.
25 Normally, a macro can be invoked using a simple command such as a
26 keystroke. The application program can be, for example, Microsoft
27 Word or Microsoft Excel.
28

1 "global environment" is an area within a storage medium that
2 is associated with a particular application program and stores
3 parameters and/or macros that are used every time said application
4 program is used. For example, the global environment for a
5 particular application program can contain text, graphics, and one
6 or more macros.
7

8
9 "local document" is a document that has been generated by an
10 application program.
11

12 "virus" is a malicious computer program that replicates
13 itself.
14

15 "macro virus" is a virus consisting of one or more macros.
16

17 "payload" is an unwanted destructive task performed by a
18 virus. For example, the payload can be reformatting a hard disk,
19 placing unwanted messages into each document created by an
20 application program, etc.
21

22 "heuristics" is a set of inexact procedures.
23

24 "publicly identified macro virus" is a macro virus that has a
25 known viral signature.
26
27
28

1 "publicly unidentified macro virus" is a macro virus that
2 cannot be identified by antivirus software using viral signature
3 matching techniques.
4

5 Detailed Description of the Preferred Embodiments

6
7 The purpose of the present invention is to detect and
8 eliminate macro viruses in a generic manner, i.e., the present
9 invention works regardless of whether the virus is a publicly
10 identified or a publicly unidentified macro virus.
11

12 The present invention uses heuristics that can determine
13 effectively whether any given code 15 contains a macro virus or
14 not, and determine exactly the set of macros that comprise the
15 virus. This is achieved by means of detection module 17 analyzing
16 code 15 for the presence of two types of instructions, as described
17 below.
18
19

20 The present invention offers the following advantages over the
21 prior art:
22

- 23 • a generic detection and repair solution for new (publicly
24 unidentified) macro viruses.
- 25
- 26 • virtually no turnaround time.
- 27
- 28

- 1 • ability to determine with an extremely high degree of confidence
- 2 that a set of macros declared to be a virus is indeed a virus.
- 3
- 4 • ability to determine the set of macros that comprise the virus,
- 5 thus providing an immediate repair solution.
- 6
- 7 • reduced workload for all personnel involved, in terms of virus
- 8 discovery, analysis, and definition creation.
- 9
- 10 • increased user satisfaction with regard to protection against
- 11 new (publicly unidentified) viruses.
- 12

13 Figure 1 illustrates a typical operating environment of the
14 present invention. A digital computer 1 comprises a processor 4
15 and memory 3. When it is to be executed, application program 5 is
16 moved into memory 3 and is operated upon by processor 4.
17 Application program 5 is any program that generates macros, for
18 example, Microsoft Word or Microsoft Excel. When it is executed,
19 application program 5 generates one or more local documents 11,
20 which are stored in storage medium or media 9 associated with
21 computer 1. For example, storage medium 9 can be a hard disk,
22 floppy disk, tape, optical disk, or any other storage medium used
23 in connection with digital computers. Each document 11 can
24 comprise text, graphics, and/or one or more macros 8. Each macro 8
25 is typically contained within a module 12. A module can contain
26
27
28

1 one or more macros 8. A user of computer 1 typically communicates
2 with application program 5 via user interface 7, which may comprise
3 a keyboard, monitor, and/or mouse.
4

5 Figure 2 shows a document 11 that has been opened by
6 application program 5. Because document 11 has been so opened, it
7 resides in memory 3, where it can be readily and quickly accessed
8 by application program 5. As stated previously, document 11 can
9 contain one or more macros 8. If one of these macros 8 is named
10 AutoOpen or a similar name, said macro 8 will execute
11 automatically. Alternatively, the macro 8 could execute upon the
12 user pressing a certain key on keyboard 7, or upon the occurrence
13 of another event.
14
15

16 Figure 2 also illustrates the presence of the global
17 environment 13 that is associated with application program 5.
18 Global environment 13 is available to the user every time he or she
19 uses application program 5, and is specific to each such
20 application program 5. Global environment 13 typically contains a
21 set of macros 8 previously established by the user, orders of
22 menus, new menu items, and preferences of the user, e.g., font
23 styles and sizes.
24
25

26 There is typically just one global environment 13 per computer
27 1 per application program 5, even if different users share the same
28

1 program 5. Global environment 13 is located within storage medium
2 10. Storage medium 10 can be the same storage medium 9 as used by
3 one or more documents 11 that have been generated by application
4 program 5. Alternatively, storage medium 10 may be distinct from
5 storage medium 9 or storage media 9. Storage medium 10 can be any
6 storage device used in conjunction with a digital computer, such as
7 a hard disk, floppy disk, tape, optical disk, etc.
8
9

10 If application program 5 is the spreadsheet software known as
11 Microsoft Excel, then global environment 13 has a typical location
12 which is the subdirectory named "XLSTART" in the directory where
13 Excel is stored (i.e., "C:/EXCEL/XLSTART"); a document 11 is called
14 a "workbook"; macros 8 reside within modules 12 in the workbook;
15 and the language in which the macros 8 are written is Visual Basic,
16 an object oriented language.
17
18

19 Figure 3 illustrates how macro viruses propagate (replicate)
20 into the global environment 13. In step 1, document 11 is opened
21 by application program 5. During step 1, document 11, including
22 all the elements contained therewithin, move from storage medium 9
23 to memory 3. In the illustrated embodiment, document 11 comprises
24 a module 12 containing a macro 8, text, and graphics. The text may
25 be, for example, a letter that the user has created previously.
26 All of these items move to memory 3.
27
28

1 Let us assume that macro 8 contains a virus. In step 2, in
2 the case where program 5 is Microsoft Excel, macro virus 8 causes
3 the entire document 11 to be moved to global environment 13.
4

5 In step 3, macro virus 8 manifests its payload, usually by
6 copying itself into another local document 11. Step 3 can be
7 precipitated every time a new document 11 is generated by
8 application program 5 or less often; for example, every time that
9 document 11 is a letter addressed to a certain individual. In any
10 event, the payload of macro 8 has a highly negative effect on
11 computer 1. For example, this payload can entail infecting certain
12 documents 11 with gibberish, reformatting a storage medium 9, 10,
13 etc.
14

15
16 Thus does macro virus 8 infect the global environment 13, and
17 from there is poised like a coiled snake ready to infect other
18 documents 11. This is because the global environment 13 is always
19 active, and thus, macro virus 8 is always active. From the newly
20 infected documents 11, this virus 8 can infect the global
21 environments 13 of all users to whom the infected documents 11 are
22 passed.
23
24

25 Figure 4 illustrates apparatus by which the present invention
26 detects and eliminates macro viruses. Code 15 is what is analyzed
27 by the present invention for the presence of macro viruses. Code
28

1 15 is adapted to be used from within computer 1, where it will be
2 coupled to the documents 11 generated by application program 5 and
3 to global environment 13. Coupled to code 15 is detection module
4 17, which is the module of the present invention that determines
5 whether a macro virus is present within code 15, based upon the
6 simultaneous presence of two conditions, as described below.
7
8 Module 17 can be implemented in hardware, firmware, and/or
9 software.
10

11 Figure 4 illustrates the special case where detection module
12 17 and code 15 are simultaneously resident within computer 1.
13 However, code 15 does not have to be resident within computer 1 in
14 order for the present invention to work; code 15 can be resident on
15 a hard disk, floppy disk, or other storage medium not currently a
16 part of computer 1.
17

18
19 Detection module 17 is coupled to user interface 7, so that
20 module 17 may announce its decisions concerning detection of macro
21 viruses to the user. Coupled to detection module 17 is repair
22 module 19, which eliminates macro viruses that have been determined
23 by detection module 17 to be present. Repair module 19 is also
24 coupled to code 15. Module 19 can be implemented in hardware,
25 firmware, and/or software.
26
27
28

1 The method of the present invention is illustrated in Figure
2
3 5. In step 51, detection module 17 begins to analyze code 15. It
4 does this by examining a first module 12 within code 15. Module 17
5 first determines whether the module 12 contains a macro 8. The
6 determination as to whether something is a macro is a conventional
7 technique in computer programming. Another (optional)
8 preprocessing step is for module 17 to determine the language in
9 which code 15 has been written. This determination is again a
10 conventional technique of computer programming. In the special
11 case where code 15 is meant to be used in association with the
12 Microsoft Excel spreadsheet application program 5, this step is not
13 necessary, because it is known that the macros 8 in Excel have to
14 be written in the Visual Basic language.
15

16
17 In step 52, module 17 determines whether the code 15 under
18 test contains instructions that cause a macro 8 to be moved to a
19 global environment 13. This is the first of two conditions that
20 module 17 uses to determine the presence of a macro virus. In the
21 case where the code 15 is written in Visual Basic, module 17
22 preferably deems that this first condition is satisfied when a
23 SaveAs command that performs this operation is present within code
24 15. If module 17 determines that the first condition is not
25 satisfied, module 17 declares at step 57 that no macro virus is
26 present within the analyzed macro 8. This declaration may be
27
28

1 passed to the user via user interface 7. At this point, there is
2 no need for module 17 to continue analyzing that macro 8, so at
3 step 56 module 17 goes to the next macro 8, and analyzes this new
4 macro 8 as before.
5

6
7 If, on the other hand, the result of step 52 is that module 17
8 has determined that the first condition for the presence of a macro
9 virus has been satisfied, module 17 flags the macro 8 under test,
10 and proceeds to the next step, step 53, where module 17 determines
11 whether the second condition for the presence of a macro virus is
12 satisfied. This condition is the presence within module 12 of
13 instructions that copy the same macro 8 that was flagged within
14 step 52 to a local document 11. In the case where the code 15 is
15 written in the Visual Basic language, the determination as to
16 whether a macro 8 is being copied to a local document 11 is
17 preferably made by detecting the presence of a Copy command that
18 performs this operation. If this second condition is not
19 satisfied, module 17 again declares at step 57 that no macro virus
20 is present.
21
22
23

24 If, on the other hand, module 17 has determined within step 53
25 that the second condition for the presence of a macro virus has
26 been satisfied, both conditions have now been satisfied, and module
27 17 declares at step 54 that a macro virus is indeed present within
28

1 module 12. This declaration can be passed to the user via user
2 interface 7. As this point, in the preferred embodiment, repair
3 module 19 eliminates the virus at step 55. This is typically done
4 by "deleting" the virus, as that term is used in conventional
5 computer programming. In the preferred embodiment, repair module
6 19 deletes the entire module 12 containing the virus.
7
8 Alternatively, module 19 deletes just the macro 8 that was flagged
9 in step 52 as the subject of a move to a global environment 13 and
10 in step 53 to be the subject of a copy to a local document 11.

11
12 Lets us now give an example to illustrate the operation of the
13 preferred embodiment. In this example, application 5 is Microsoft
14 Excel and module 12 comprises three lines of Visual Basic code as
15 follows:
16

```
17        start = Application.StartupPath  
18        newname = ActiveWorkbook.Name  
19        Workbooks(newname).SaveAs filename:=start & "/" &  
20        "main.xls"
```

21 The first two lines of code constitute the initialization of
22 variables named "start" and "newname", respectively. The third
23 line of code is a command line. "Workbooks" is the set of all
24 workbooks that are currently open. Initially it is a null set,
25 because no documents 11 are open the first time Excel is opened.
26 The presence of the SaveAs command means that something called
27 Workbooks(newname) is being saved someplace. "filename" is a
28

1 parameter to the SaveAs command. We know that is a parameter
2 because of the ":= " following it. The item being saved is saved to
3 the location "start/main.xls". Note the presence of the two
4 concatenation operators "&". The presence of these concatenation
5 operators may well indicate that the author of the virus attempted
6 to disguise the presence of the virus by breaking up the
7 destination location of the SaveAs into three parts.

8
9 "Application.StartupPath" evaluates to Excel's startup directory,
10 which is its global environment 13. Assuming that this location is
11 "C:/EXCEL/XLSTART", when we substitute the value of
12 "Application.StartupPath" for the variable "start", we see that the
13 item to be saved is saved in the location
14 "C:/EXCEL/XLSTART/main.xls". main.xls is an arbitrary name given
15 by the author of the virus. We know that Application.StartupPath
16 is the location of the startup directory within a global
17 environment 13. Global environment 13 also contains an alternative
18 startup directory that can be referred to from within a Visual
19 Basic macro using Application.AltStartupPath.

20
21 Making the substitution of variables, the item being saved to
22 the global environment 13 is a local document 11 called
23 "ActiveWorkbook.Name". In the context of this module 12, this
24 active workbook 11 is the one containing the module 12, and thus
25
26
27
28

1 this module 12 (which is known to contain a macro 8) is saved to
2 the global environment 13.

3
4 Thus, module 12 contains instructions causing a macro 8 to be
5 moved to a global environment 13. Therefore, the first condition
6 (step 52) for the presence of a macro virus is satisfied.
7

8 In our example, let us assume that module 12 also contains the
9 following code:
10

```
11         papy_name = ActiveWorkbook.Name  
12         Workbooks("MAIN.XLS").Sheets("junior").Copy  
13         before:=Workbooks(papy_name).Sheets(1)
```

14 In the Visual Basic syntax, something to the left of an "=" is
15 a variable.

16 "Name" is a user-given name.
17

18 "junior" is the name of the module 12 that contains the virus.
19 We know that the module 12 containing the three lines of code that
20 satisfied condition one is in "junior" (part of main.xls), because
21 we assume that module 17 has noted that "junior" is the name of the
22 module 12 containing this code. Alternatively, we can use other
23 means to determine where the suspicious code 15 is going to be
24 sent, e.g., we can determine whether there is some other code
25 renaming "junior" as "senior".
26
27
28

1 In this example, "before" means "the location just before".
2 Here the syntax requires an "after" or a "before". In the second
3 line of this code, the module named "junior" from the globally
4 installed workbook 11 named "main.xls" is copied to the active
5 workbook 11, which, at the time of execution of the above two
6 lines, is the workbook 11 currently being edited by the user.
7

8
9 Thus, the same module 12 that was flagged in step 52 as being
10 saved into the global environment 13 is now uncovered to be copied
11 into a local document (workbook 11). Therefore, module 17
12 determines that condition two is satisfied, and therefore declares
13 that this module 12 contains a macro virus.
14

15 In order for the implementation of this invention to be
16 robust, and thus able to detect the greatest number of viruses, it
17 is desirable, although not necessary, for detection module 17 to
18 have one or more of the following four enhancement features. Each
19 of these features slows down the operation of module 17. However,
20 one can implement each of these features with extreme robustness or
21 minimum robustness, depending upon the degree of robustness
22 expected to be desirable. The factors that go into this
23 determination include memory 3 constraints, execution time
24 constraints, and expected real world scenarios.
25
26
27
28

1 The four enhancement features are:

2
3 1. The ability to understand and handle string concatenation
4 operations in the analyzed modules 12. For example, any string in
5 Visual Basic can be concatenated. The reason for this first
6 enhancement feature is that, for example, the virus designer could
7 have referred to "junior" as "jun" & "ior". In this case, it is
8 desirable for module 17 to recognize this as "junior". Module 17
9 will so recognize this as "junior" if enhancement feature 1 is
10 present.
11

12
13 2. The second enhancement feature is the ability for module 17
14 to understand and handle variable assignment operations in order to
15 trace the values of variables. This is sometimes referred to as
16 variables being "proxied". For example, the virus designer could
17 have written the code to be:
18

19
20 papy_name=ActiveWorkbook.name
21 a="jun"
22 b="ior"
23 c=a&b
24 Workbooks("MAIN.XLS").Sheets(c).Copy
25 before:=Workbooks(papy_name).Sheets(1)

26 If we want to detect "junior" in this case, we need to make
27 all of these variable substitutions. Since there is a constraint
28 on the size of memory 3, and on the amount of time that can be
used, we might limit module 17 to where it will perform only an

1 arbitrary number (for example 16) of substitutions. This is
2 because it is expected that the amount of variable usage is small
3 in most cases, and we accept this tradeoff between robustness and
4 speed/memory constraints.
5

6
7 3. The third enhancement feature is for module 17 to have the
8 ability to trace the value of parameter values through function
9 calls (subroutine calls). For example, suppose that code 15
10 contains:

```
11  
12 sub CopyIt(SheetName)  
13     papy_name = ActiveWorkbook.Name  
14     Workbooks("MAIN.XLS").Sheets(SheetName).Copy  
15     before:=workbooks(papy_name) Sheets(1)  
16 End Sub  
17  
18 Sub InfectDocument  
19     sName = "junior"  
20     CopyIt(sName)  
21 End Sub
```

22 In this example, the value of the parameter variable
23 SheetName is being passed to the subroutine CopyIt from another
24 subroutine (function) called InfectDocument, in an attempt by the
25 virus designer to obfuscate what is happening.
26

27
28 4. The fourth enhancement feature is giving module 17 the
ability to understand and handle the object model of Visual Basic,
e.g., object/subobject hierarchy parsing and understanding. For

1 example, a straightforward way of initializing the variable "start"
2 would be to write:

3
4 start=Application.StartupPath

5 However, if the virus designer wanted to obfuscate what he or
6 she was doing, said designer might write:

7
8 app=Application
9 start=app.StartupPath

10 In this case, "Application" is an object. If the fourth
11 enhancement feature is present, module 17 has the ability to handle
12 substituted object names.

13
14 The above description is included to illustrate the operation
15 of the preferred embodiments and it is not meant to limit the scope
16 of the invention. The scope of the invention is to be limited only
17 by the following claims. From the above discussion, many
18 variations will be apparent to one skilled in the art that would
19 yet be encompassed by the spirit and scope of the present
20 invention.
21

22
23 What is claimed is:
24
25
26
27
28